

S. M. Sadegh Tabatabaei Yazdi and Lara Dolecek  
EE Dept. UCLA, {sadegh,dolecek}@ee.ucla.edu

**Abstract**—In this paper, we consider a synchronization problem between nodes  $A$  and  $B$  that are connected through a two-way communication channel. Node  $A$  contains a binary file  $X$  of length  $n$  and node  $B$  contains a binary file  $Y$  that is generated by randomly deleting bits from  $X$ , by a small deletion rate  $\beta$ . The location of deleted bits is not known to either node  $A$  or node  $B$ . We offer a deterministic synchronization scheme between nodes  $A$  and  $B$  that needs  $O(n\beta \log \frac{1}{\beta})$ <sup>12</sup> transmissions and reconstructs  $X$  at node  $B$  with probability of error that is exponentially low in the size of  $X$ . The rate of our scheme matches the optimal rate for this channel. Our scheme can be extended to other editing models, e.g., insertions, repetitions or replacements, as long as the rate of editing is small.

**Keywords:** Two-way communication, deletion channel, synchronization, edits, coding for synchronization.

## I. INTRODUCTION

Consider two nodes  $A$  and  $B$  that respectively hold files  $X$  and  $Y$ , where file  $Y$  can be derived from file  $X$  by some deletions. For instance let

$$X = \underset{D}{00} \underset{D}{101} \underset{DD}{1000} \underset{D}{10101} \underset{D}{11}, \text{ and}$$

$$Y = 00010001011.$$

Here  $Y$  is derived from  $X$  by 5 deletions, where deleted bits are denoted by  $D$ . We call  $Y$  a deleted version of  $X$ . Suppose that the locations of deleted bits are *unknown* to both nodes. In this paper we are interested in the following question:

- What is the optimal transmission protocol for *synchronizing* the content of node  $B$  with the content of node  $A$ , i.e., how to reconstruct an estimate of file  $X$  at node  $B$ ?

By way of optimality, we are mainly concerned with the number of transmitted symbols between two nodes and the complexity of implementing the protocol at nodes  $A$  and  $B$ . Also, as usual, we desire the reconstructed estimate of  $X$  at node  $B$  to have symbol error probability that is exponentially small in the size of  $X$ .

Synchronization from deletions is a special case of a more general synchronization problem where file  $Y$  can be derived from  $X$  by a sequence of edits. An edit can refer to either deletion of a bit from file  $X$  or insertion of a new bit within  $X$ . File synchronization from random edits is the subject of many practical applications. Over the web, file updating is

The paper is presented in part at the *IEEE 7th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Aug. 2012.

<sup>1</sup> $f(n) \in O(g(n))$  if there exist positive constants  $C$  and  $n_0$  such that  $|f(n)| \leq Cg(n)$  for  $n \geq n_0$ .

<sup>2</sup>All logarithms in this paper are in base 2.

an application where a user or a server needs to synchronize its outdated version of a file with a newer version. The new updates of a file can usually be modeled as random edits of its content. As another example, consider a search engine that constantly updates its database in order to reflect the latest changes to the content of websites. Here as well, changes can be modeled by random edits to the content of websites. Another area of application is in distributed storage networks where several backup nodes store the same content and need to be regularly synchronized together. Mis-synchronization in storage devices can be due to mis-synchronized clock speeds of read and write heads of hard drives or crashes in random parts of the hard drive.

### A. Previous Work

There has been a large body of research on synchronization from edits. In [1], Varshamov and Tenegolts offered a coding scheme for recovery from one asymmetric error. Soon thereafter, Levenshtein [2] showed that the scheme of Varshamov and Tenegolts can be used for synchronization from one deletion or one insertion. In [3], Orlitsky proved several bounds on the minimum number of transmitted bits under a restricted number of communication rounds. He further showed that if the number of edits is known to the nodes  $A$  and  $B$ , then there is a protocol that can asymptotically achieve the number of transmitted bits identical to the case where node  $A$  has access to file  $Y$ .

While the results of [3] are nonconstructive, several researchers have provided explicit code constructions. Let  $n$  denote the length of file  $X$ . For  $\delta$  number of edits, Schwarz *et al.* [4] devised a protocol based on hash functions that needs  $O(\delta \log n \log \frac{n}{\delta})$  transmitted bits. Cormode *et al.* [5] offered an  $\epsilon$ -error protocol with  $c(\epsilon)\delta \log^3 n$  total transmitted bits, where  $c(\epsilon)$  is a constant that depends on the error  $\epsilon$ . For the same setting, Evfimievski [6] devised a protocol with the number of transmitted bits that is a polynomial in  $\log n$ ,  $\log \frac{1}{\epsilon}$ , and  $\delta$ . For an unknown, fixed number of edits  $\delta$ , Orlitsky and Viswanathan [7] showed that the  $\epsilon$ -error optimal protocol needs at most  $\delta \log n + \log \frac{1}{\epsilon}$  transmitted bits. They also provided an explicit synchronization protocol that needs  $2\delta \log n (\log n + \log \log n + \log \frac{1}{\epsilon} + \log \delta)$  transmitted bits. More recently, Venkataramanan *et al.* [8] offered a synchronization scheme that can correct  $\delta = o(\frac{n}{\log n})^3$  edits with  $(4c+1)\delta \log n$  transmitted bits from node  $A$  to node  $B$  and

<sup>3</sup> $f(n) \in o(g(n))$  if for every  $\varepsilon > 0$  there exists  $n_0$  such that  $|f(n)| \leq \varepsilon|g(n)|$  for  $n \geq n_0$ .

$10(\delta - 1)$  transmitted bits from node  $B$  to node  $A$  for any positive integer  $c$ . The error of the reconstruction is at most  $\frac{d \log n}{n^c}$  where  $d$  is the number of deleted bits in  $X$ , out of  $\delta$  total edits.

In practice, RSYNC [9] is a popular UNIX application for synchronizing between edited files. The RSYNC method can be in general very inefficient and the number of transmitted bits can be exponentially larger than the optimal number. There have been many improvements over the baseline approach. For example Suel *et al.*, [10] proposed a protocol that sometimes can save up to 50% of bandwidth over RSYNC. There are also more specialized synchronization tools, such as VSYNC [11], which synchronizes between video files.

### B. Our Contribution

While most of the previous work has concentrated on synchronizing from a fixed number of edits between the two files  $X$  and  $Y$ , in this paper we are interested in a more practical scenario, which is synchronizing from a *fixed rate* of edits between the two files. In this paper we only study synchronization from deletions and will discuss possible extensions to the more general case of deletions and insertions. More specifically, we consider synchronization between node  $A$  and node  $B$  where node  $A$  has a binary string  $X$  that is generated by an i.i.d Bernoulli process of parameter  $\frac{1}{2}$ . Node  $B$  has a binary string  $Y$  that is generated from  $X$  by randomly and independently deleting bits of  $X$  with probability  $\beta$  that is very small. We are interested in an optimal transmission protocol for synchronizing between nodes  $A$  and  $B$  when  $n$ , the length of  $X$ , is large.

In order to evaluate a lower bound on the optimal number of transmitted bits between nodes  $A$  and  $B$ , suppose that node  $A$  has access to string  $Y$ . Then, the optimal number of transmitted bits to node  $B$ , needed for reconstructing  $X$  is  $H(X|Y)$  which is the conditional entropy of string  $X$  given string  $Y$ . Ma *et al.* [12] considered a more general set-up where the deletion pattern follows a stationary Markov chain. By applying the result of [12] to our model, for small values of  $\beta$ , the entropy  $H(X|Y)$  can be estimated as follows

$$H(X|Y) = n(\beta \log \frac{1}{\beta} + O(\beta)).^4 \quad (1)$$

Therefore, any synchronization protocol needs at least  $n\beta \log \frac{1}{\beta}$  transmitted bits. Paper [12] further uses tools from a well studied problem of source coding with side information [13], [14] to show that there exists a randomized synchronization protocol on a *one-way* channel that asymptotically needs  $H(X|Y)$  transmitted bits. However, [12] does not offer any explicit, deterministic construction for the synchronization protocol. Notice that the most efficient previous constructions (e.g., [8]) for a *fixed* number of edits  $\delta$  require  $O(\delta \log n)$  transmitted bits between  $A$  and  $B$ . A naïve application of such results to our setup would require  $O(n\beta \log n)$  transmitted bits between  $A$  and  $B$  for large  $n$ , which is clearly far from being optimal.

<sup>4</sup> $f(\beta) \in O(g(\beta))$  if there exists positive constants  $C$  and  $\beta_0$  such that  $|f(\beta)| \leq Cg(\beta)$  for  $\beta \leq \beta_0$ .

In this paper, we offer the first explicit and deterministic construction of a protocol for synchronizing from a small rate of deletions on a two-way channel. The protocol is optimal within a constant multiplicative factor and needs  $O(n\beta \log \frac{1}{\beta})$  transmitted bits. Furthermore, we demonstrate that the error probability of synchronization at node  $B$  is exponentially small in  $n$ . Finally, we show that our scheme needs a running time that is at most  $O(n^4\beta^6)$ .

The rest of the paper is organized as follows. In Section II we present the problem setting and the main result along with a sketch of our synchronization scheme. In Section III we present the mathematical details of our synchronization protocol and the proof of the main result in the paper. Section IV discusses practical implications of our protocol for low-complexity synchronization algorithms and Section V includes concluding remarks and directions for possible extensions.

## II. PROBLEM SETTING AND THE MAIN RESULT

### A. Preliminaries

We represent a binary string  $Z$  of length  $\ell$  by  $Z = Z(1), Z(2), \dots, Z(\ell)$ . For  $1 \leq i \leq j \leq \ell$ ,  $Z(i, j)$  denotes the substring  $Z(i), Z(i+1), \dots, Z(j)$  of  $Z$ . If  $Z_1$  is a string of length  $\ell_1$  and  $Z_2$  is a string of length  $\ell_2$ , we denote by  $Z_1, Z_2$  the string of length  $\ell_1 + \ell_2$  obtained by concatenation of  $Z_1$  and  $Z_2$ . For a string  $Z$  we let  $|Z|$  denote the length of  $Z$ .

A *deletion channel* is a channel that deletes a subset of the bits of the input string. A deletion channel is characterized by a deletion pattern  $D$  which is a binary string of the same length as the input string. Let  $X$  be the input to the deletion channel with deletion pattern  $D$  and  $Y$  be the output of the channel. The deletion channel deletes bit  $X(i)$  from the input  $X$  if  $D(i) = 1$  and transmits the bit  $X(i)$  if  $D(i) = 0$ . For example, the output of a deletion channel with input  $X = 101$  and deletion pattern  $D = 010$ , is  $Y = 11$ .

Corresponding to the deletion pattern  $D$ , we define a function  $f_D$  which maps the indices of bits in the input string, to their corresponding indices in the output string. If for index  $i$ ,  $D(i) = 0$ , then  $f_D(i) = i - \sum_{j < i} D(j)$ , and if  $D(i) = 1$ , then  $f_D(i) = f_D(i')$  where  $i'$  is the largest index, smaller than  $i$ , for which  $D(i') = 0$ .

### B. The Main Result

Suppose that node  $A$  contains a file that is represented by a binary string  $X$  of length  $n$ . Let node  $B$  contain a file  $Y$  of length  $m$  that is the output of a deletion channel with input  $X$  and deletion pattern  $D$ . We assume that the deletion pattern is unknown to nodes  $A$  and  $B$ . Suppose that the source file  $X$  is generated by an i.i.d Bernoulli source of parameter  $\frac{1}{2}$  and that the deletion channel deletes bits of  $X$  independently and with probability  $\beta \ll 1$ . We are interested in a synchronization protocol on a two-way, error-free channel between nodes  $A$  and  $B$  so that node  $B$  can recover string  $X$  from string  $Y$  with a small probability of error at the end of the communication session. Our main contribution in this paper is proving the following theorem.

**Theorem 1.** *There exists a deterministic synchronization protocol between nodes A and B on a two-way channel, that on average transmits  $O(n\beta \log \frac{1}{\beta})$  bits and generates an estimate  $\hat{X} = \hat{X}(1), \dots, \hat{X}(n)$  of  $X$  at node B, such that  $\Pr \{ \hat{X}(i) \neq X(i) \} \leq 2^{-\Omega(n)}$  for every  $1 \leq i \leq n$ .<sup>5</sup>*

We prove the theorem by explicitly constructing a synchronization protocol. Next, we provide an overview of our synchronization protocol and prove its optimality.

### C. Synchronization Protocol

Recall that node B has string  $Y$  which is a deleted version of string  $X$ . We next explain a synchronization protocol that enables node B to reconstruct an estimate of string  $X$  with a small probability of error. The synchronization protocol has three main steps as illustrated in Figure 1. Each step is performed by a module at node B that has a two-way communication link to node A. The three modules work in serial, such that the input to the first module is string  $Y$  and the output of the last module is the estimate  $\hat{X}$  of string  $X$ . Suppose that  $X$  is divided into substrings as follows

$$X = T_1, S_1, T_2, S_2, \dots, T_{k-1}, S_{k-1}, T_k,$$

where  $|S_i| = L_S$  and  $|T_i| = L_T$ . Substrings  $S_1, \dots, S_{k-1}$  are called *pivot strings* and substrings  $T_1, \dots, T_k$  are called *data strings*. We choose  $L_T = \frac{1}{\beta}$  and  $L_S = O(\log \frac{1}{\beta})$  and both nodes A and node B know the exact values of  $L_T$  and  $L_S$ . Note that the length of a pivot string is much smaller than the length of a data string. We will determine the exact value of  $L_S$  later during our analysis.

- 1) The first step of the decoding process is performed by the *matching module* at node B. In this step, node A sends pivot strings  $S_i$ ,  $1 \leq i \leq k-1$ , to node B. Upon receiving the pivots, the matching module attempts to figure out the positions of pivots in  $Y$  by finding the *exact copies* of  $S_i$ 's within  $Y$ . Due to possible deletions within  $S_i$ 's, the matching module is able to find the exact matches for only a subset of  $S_i$ 's.<sup>6</sup> Suppose that the matching module finds matches for  $S_{i_1}, \dots, S_{i_{k'-1}}$  where  $k' \leq k$ . Based on the position of matched  $S_i$ 's, the matching module divides  $Y$  into substrings as follows and sends it to the next module,

$$Y = \bar{P}_1, S_{i_1}, \bar{P}_2, S_{i_2}, \dots, \bar{P}_{k'-1}, S_{i_{k'-1}}, \bar{P}_{k'}, \quad (2)$$

where  $\bar{P}_j$  denotes the substring between matched pivots  $S_{i_{j-1}}$  and  $S_{i_j}$  in  $Y$ .

- 2) The next step is performed by the *deletion recovery module* at node B. After receiving the divided  $Y$  from the matching module, the deletion recovery module sends the indices  $\{i_1, \dots, i_{k'-1}\}$  of the matched pivots in  $Y$  to node A. Upon receiving the indices, node A divides  $X$  into substrings as follows:

$$X = P_1, S_{i_1}, P_2, S_{i_2}, \dots, P_{k'-1}, S_{i_{k'-1}}, P_{k'}, \quad (3)$$

<sup>5</sup>  $f(n) \in \Omega(g(n))$  if there exist positive constants  $C$  and  $n_0$  such that  $f(n) \geq Cg(n)$  for  $n \geq n_0$ .

<sup>6</sup> We will explain later the other possible cases when there are multiple matches for a pivot but an error is made by detecting a match that is not due to the original pivot.

where  $P_j$  denotes the substring between pivots  $S_{i_{j-1}}$  and  $S_{i_j}$  in  $X$  and can be written as follows:

$$P_j = T_{i_{j-1}+1}, S_{i_{j-1}+1}, \dots, S_{i_j-1}, T_{i_j}.$$

Notice that if  $S_{i_{j-1}}$  and  $S_{i_j}$  are matched correctly in  $Y$ , then  $\bar{P}_j$  can be derived from  $P_j$  by some sequence of deletions. In this step, nodes A and B use the synchronization protocol of Venkataramanan *et al.*, [8] with parameter  $c = 3$ <sup>7</sup> to recover from deleted bits of  $\bar{P}_j$  and to form an estimate of  $P_j$  for each  $1 \leq j \leq k$ . Let us denote by  $\tilde{P}_j$  the estimate of  $P_j$  at the output of the deletion recovery module. Notice that  $\tilde{P}_j$  has the same length as  $P_j$ . At the end of this step, the deletion recovery module forwards the string

$$\tilde{X} = \tilde{P}_1, S_{i_1}, \tilde{P}_2, S_{i_2}, \dots, \tilde{P}_{k'-1}, S_{i_{k'-1}}, \tilde{P}_{k'} \quad (4)$$

as an estimate of  $X$  to the next module.

- 3) At this step, the *LDPC decoder module* at node B, recovers from the errors made by the first two steps. At the first step of the protocol, due to a potential existence of multiple copies of each  $S_i$  within  $Y$ , the matching module may erroneously match  $S_i$  at a wrong place. Suppose  $S_{i_j}$  is a pivot that the matching module has matched at a wrong place. Then,  $\bar{P}_j$  and  $\bar{P}_{j+1}$  may not be realizable by deleting subsets of bits from  $P_j$  and  $P_{j+1}$  respectively. As a result, after the deletion recovery module,  $\tilde{P}_j$  and  $\tilde{P}_{j+1}$  may be different from  $P_j$  and  $P_{j+1}$ , respectively. Furthermore, even if the matching module has matched pivots  $S_{i_{j-1}}$  and  $S_{i_j}$  correctly in  $Y$  and  $\bar{P}_j$  is a deleted version of  $P_j$ , the protocol of Venkataramanan *et al.* [8], used in deletion recovery module, could introduce additional errors. Suppose that the total error of the first two synchronization modules is bounded by  $\zeta$ ,

$$\Pr \{ \tilde{P}_j \neq P_j \} \leq \zeta.$$

We notice that the output of the deletion recovery module,  $\tilde{X}$ , is in synchronization with  $X$ , in the sense that  $|\tilde{P}_j| = |P_j|$  for each  $1 \leq j \leq k'$  and hence  $\tilde{X}(i)$  is the estimate of  $X(i)$  for each index  $1 \leq i \leq n$ . Since the error rate for substrings  $\tilde{P}_j$ ,  $1 \leq j \leq k'$ , is an upper bound for the bit error rate in  $X$ , we find that

$$\Pr \{ \tilde{X}(i) \neq X(i) \} \leq \zeta.$$

As a result,  $\tilde{X}$  can be modeled as an output of a Binary Symmetric Channel (BSC) with the crossover probability at most  $\zeta$ . To recover from errors of  $\tilde{X}$  we then use a powerful additive-error correction code. Our choice is an LDPC decoder which receives parity check bits of a systematic LDPC code [15]. If node A sends a sufficient number of parity check bits to the LDPC decoder module, as shown in [16], the output of the decoder will be a string  $\hat{X}$  with

$$\Pr \{ \hat{X}(i) \neq X(i) \} \leq 2^{-\Omega(n)},$$

<sup>7</sup> Here,  $c$  is an arbitrary constant that defines the tradeoff between complexity of the protocol and the error in the output of the decoder.

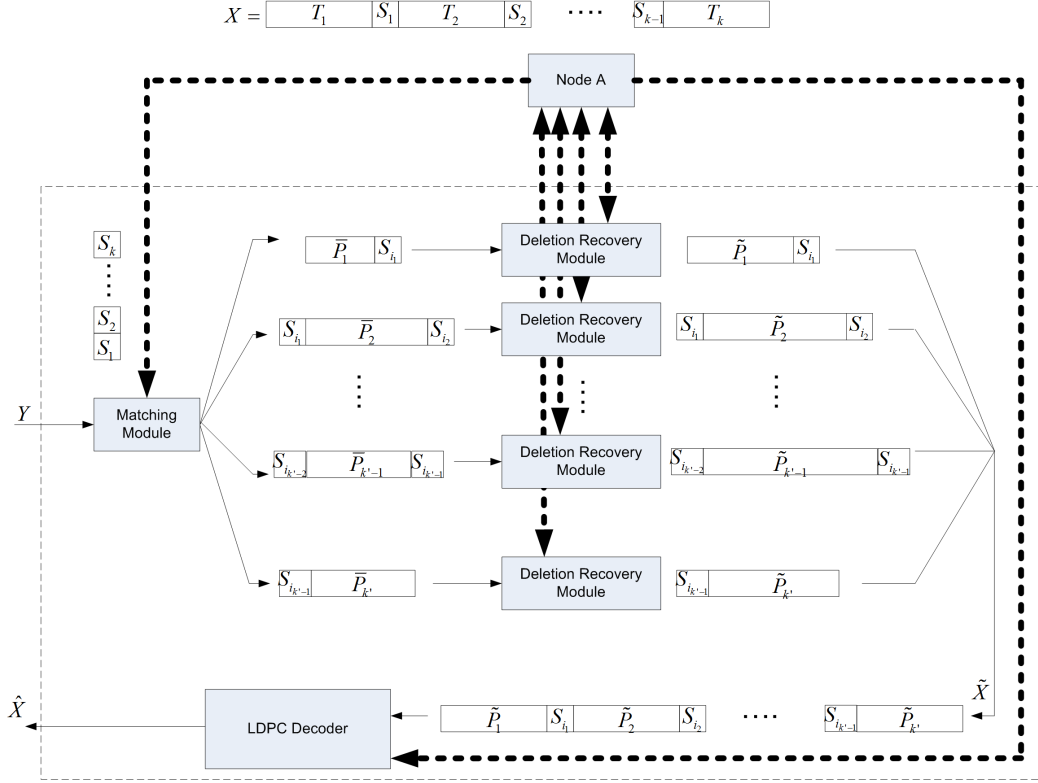


Figure 1. Illustration of the synchronization protocol.

as previously stated in Theorem 1.

Next, we wish to estimate the total number of transmitted bits used by our synchronization protocol. We first establish a measure of the performance of the matching module of the decoder.

**Theorem 2.** For  $L_S \geq 11 + 2 \log \frac{1}{\beta}$ , there exists a matching module that, with probability  $1 - 2^{-\Omega(n)}$ , matches a subset  $\{S_{i_1}, \dots, S_{i_{k'}}\}$  of pivots  $\{S_1, \dots, S_{k-1}\}$  with  $k' = (1 - L_S \beta + 2\beta + o(\beta))k$ <sup>8</sup> such that the probability of error in matching  $S_{i_j}$  is at most  $\beta + o(\beta)$ .

We devote Section III to proving this theorem. For the rest of our argument we set  $L_S = 11 + 2 \log \frac{1}{\beta}$ , which is the minimum value of  $L_S$  required by Theorem 2.

Next we use Theorem 2 to estimate the total number of transmitted bits needed by the synchronization protocol.

**Lemma 1.** On average, the total number of transmitted bits of the synchronization protocol is  $O(n\beta \log \frac{1}{\beta})$ .

*Proof:* First notice that  $k = \frac{n+L_S}{L_T+L_S} \approx n\beta$ . The number of transmitted bits in the first step is

$$(k-1)L_S \approx 11n\beta + 2n\beta \log \frac{1}{\beta} = O(n\beta \log \frac{1}{\beta}).$$

At the second step, the protocol of Venkataramanan *et al.* [8] for the recovery from deletions within each  $P_j$ ,  $1 \leq j \leq k'$

<sup>8</sup>  $f(\beta) \in o(g(\beta))$  if for every  $\varepsilon > 0$  there exists  $\beta_0$  such that  $|f(\beta)| \leq \varepsilon|g(\beta)|$  for  $\beta \leq \beta_0$ .

with parameter  $c = 3$  needs  $13\delta_j \log |P_j| + 10(\delta_j - 1)$  transmitted bits, where  $\delta_j := |P_j| - |\tilde{P}_j|$  is the number of deleted bits in  $P_j$ . Therefore, the average number of transmitted bits in the second step is less than

$$\mathbb{E} \left[ \sum_{j=1}^{k'} (13\delta_j \log |P_j| + 10\delta_j) \right].$$

Notice that  $\sum_{j=1}^{k'} \delta_j$  is the total number of deleted bits from  $X$  and is on average  $n\beta$  (recall that we assumed that no deletions occurred in the matched pivots).

In Appendix I we show that  $\mathbb{E}[\delta_j \log |P_j|] \leq 16 + 8 \log \frac{1}{\beta}$ . Therefore, the average number of transmitted bits in the deletion recovery module is upper bounded by

$$\begin{aligned} k' 13(16 + 8 \log \frac{1}{\beta}) + 10n\beta &\leq n\beta 13(16 + 8 \log \frac{1}{\beta}) + 10n\beta \\ &= O(n\beta \log \frac{1}{\beta}), \end{aligned}$$

where we used the inequality  $k' \leq k \approx n\beta$ .

For the last step, we would like to estimate the error  $\zeta$  in  $\tilde{P}_j$ . By Theorem 2, the error probability in matching  $S_{i_{j-1}}$  and  $S_{i_j}$  is at most  $\beta$  each. Since  $\tilde{P}_j$  is the common neighbor of  $S_{i_{j-1}}$  and  $S_{i_j}$ , with probability at most  $2\beta$ , the string  $\tilde{P}_j$  is not a deleted version of  $P_j$ . Furthermore, the error in the protocol of Venkataramanan *et al.* [8] for  $c = 3$ , is upper bounded by  $\frac{\delta_j \log |P_j|}{|P_j|^3}$ . Since  $\mathbb{E}[\delta_j] = \beta L_T = 1$  and also  $|P_j| = L_T = \frac{1}{\beta}$ , the average probability of error by the protocol of Venkataramanan *et al.*, is upperbounded

by  $\beta^3 \log \frac{1}{\beta} = o(\beta)$ . Counting the error from matching module, we have  $\Pr\{\tilde{P}_j \neq P_j\} \leq 2\beta + o(\beta)$ , and therefore  $\Pr\{\tilde{X}(i) \neq X(i)\} \leq 2\beta + o(\beta)$ .

In order to recover from errors induced by a BSC with crossover probability of at most  $2\beta + o(\beta)$ , node  $A$  needs to send  $nH(2\beta + o(\beta)) = O(n\beta \log \frac{1}{\beta})$ <sup>9</sup> parity check bits to node  $B$ .

The average number of transmitted bits in all three steps of the protocol is  $O(n\beta \log \frac{1}{\beta})$ . Therefore the average number of transmitted bits by the algorithm is  $O(n\beta \log \frac{1}{\beta})$ . ■

In the next section we prove Theorem 2.

### III. PROOF OF THEOREM 2

In this section, we propose a construction of a matching module such that for  $L_S \geq 11 + 2 \log \frac{1}{\beta}$  with probability  $1 - 2^{-\Omega(n)}$  the module matches  $k'$  pivots, out of which at most  $\beta k$  pivots are matched erroneously. Since  $\beta k = (\beta + o(\beta))k'$ , our construction implies an error of  $\beta + o(\beta)$  in matching the pivots which is equivalent to the statement of Theorem 2.

We will frequently use the following concentration theorem in our argument:

**Theorem 3** (Hoeffding [17]). *Let  $p_0$  be the probability that a biased coin shows heads. Then for every  $\varepsilon > 0$ , the probability that  $N$  tosses of the coin yield a number of heads between  $(p_0 - \varepsilon)N$  and  $(p_0 + \varepsilon)N$  is at least  $1 - 2e^{-2\varepsilon^2 N}$ .*

We will occasionally need a stronger version of the previous theorem:

**Theorem 4** (Hoeffding [17]). *Let  $z_1, \dots, z_N$  be i.i.d random variables with expected value  $M$  that take values in an interval of length  $I$ . Then, for every  $\varepsilon > 0$ , the following holds*

$$\Pr\left\{\left|\sum_{i=1}^N z_i - NM\right| \geq \varepsilon N\right\} \leq 2 \exp\left(-\frac{2\varepsilon^2 N}{I^2}\right).$$

Recall that the string  $X$  is partitioned into substrings as  $X = T_1, S_1, \dots, T_{k-1}, S_{k-1}, T_k$ , where  $|T_i| = L_T$  and  $|S_i| = L_S$ . In our setup  $L_T = \frac{1}{\beta}$ ,  $L_S = O(\log \frac{1}{\beta})$ , and  $k \approx n\beta$ . Let us denote the index of the first bit of  $S_i$  in  $X$  by  $\tilde{s}_i$  and the index of the last bit of  $S_i$  in  $X$  by  $\hat{s}_i$ . Similarly, the first and last indices of  $T_i$  are denoted by  $\tilde{t}_i$  and  $\hat{t}_i$ . Therefore,  $X(\tilde{s}_i, \hat{s}_i) = S_i$  and  $X(\tilde{t}_i, \hat{t}_i) = T_i$ .

The task of the matching module is to find “right matches” of  $S_i$ ’s within string  $Y$ . Next, we formalize the notion of right and wrong matches for a pivot  $S_i$ .

#### A. Right and Wrong Matches

Consider the substring  $D(\tilde{s}_i, \hat{s}_i)$  which is a part of the deletion pattern  $D$  that acts on the pivot  $S_i$ . We consider the following cases:

- $D(\tilde{s}_i, \hat{s}_i)$  is the all zeros vector: There is no deletion within  $S_i$ . In this case we call the copy of  $S_i$  between

<sup>9</sup>Here and in the reminder of the paper we use  $H(\cdot)$  to refer to the binary entropy function defined as follows:  $H(t) = t \log \frac{1}{t} + (1-t) \log \frac{1}{1-t}$  for  $0 < t < 1$ .

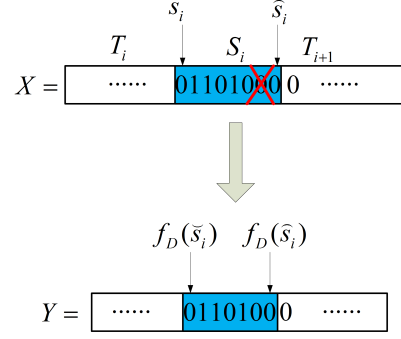


Figure 2. Illustration of a matched  $S_i$  with one deletion.

indices  $f_D(\tilde{s}_i)$  and  $f_D(\hat{s}_i)$  of  $Y$  the *right match* of  $S_i$ . All other copies of  $S_i$  in  $Y$  are considered *wrong matches* of  $S_i$ .

- $D(\tilde{s}_i, \hat{s}_i)$  has one nonzero element: There is one deletion within  $S_i$ . In this case, if there is a copy of  $S_i$  in  $Y$  that begins at  $f_D(\tilde{s}_i)$  or ends at  $f_D(\hat{s}_i)$  then we call it a right match of  $S_i$  and all other copies of  $S_i$  are called wrong matches of  $S_i$ . If there is no such copy of  $S_i$  within  $Y$ , then all copies of  $S_i$  within  $Y$  are called wrong matches. Notice that in this case there are possibly two right matches for  $S_i$ . For instance let  $S_i = 000$  and let the immediate undeleted bits before and after  $S_i$  be zero. Then it is easy to verify that after one deletion within  $S_i$ , there is a copy of  $S_i$  starting at  $f_D(\tilde{s}_i)$  in  $Y$  and there is another copy of  $S_i$  ending at  $f_D(\hat{s}_i)$  in  $Y$ .
- $D(\tilde{s}_i, \hat{s}_i)$  has more than one nonzero element: There are more than one deletions within  $S_i$ . In this case all copies of  $S_i$  within  $Y$  are considered wrong matches.

While the definition of right and wrong matches is natural for the case of no deletion within  $S_i$ , we next explain the reason behind the definition for the case with deletions within  $S_i$ . Consider the illustration in Figure 2 where  $S_i = 01101000$ . Assume the penultimate bit is deleted from it. Suppose that the bit right after  $S_i$  is 0. Notice that even with the deleted bit, a copy of  $S_i$  appears in  $Y$ , starting at  $f_D(\tilde{s}_i)$ . This copy of  $S_i$  is called a right match. The reason is that the resulting string  $Y$  is the same as in the case where there is no deletion within  $S_i$  and instead the 0 after  $S_i$  is deleted in  $X$ . In other words, here we can “move” the deletion from  $S_i$  to the substring  $T_{i+1}$  without changing  $Y$ .

Although a similar scenario may happen when there are more than one deletions within  $S_i$ , i.e., we might be able to move the deleted bits from  $S_i$  to the neighboring data strings without changing the resulting  $Y$ , since the probability of these cases is very small (the exact statement will follow), our analysis conservatively counts those matches as wrong matches.

Next, we analyze the probability of occurrence of right matches for  $S_i$ :

**Lemma 2.** *With probability  $1 - \beta L_S + o(\beta)$ ,  $S_i$  has no deletions and there is a right match for  $S_i$  within  $Y$ .*

*Proof:* With probability  $(1 - \beta)^{L_S}$  no bit is deleted from  $S_i$ . For  $L_S = O(\log \frac{1}{\beta})$  we have

$$(1 - \beta)^{L_S} = 1 - L_S\beta + o(\beta).$$

■

**Lemma 3.** *With probability  $2\beta + o(\beta)$  there is one deletion within  $S_i$  and there is a right match for  $S_i$  within  $Y$ .*

*Proof:* Fix  $h$  as the place of the deleted bit out of  $L_S$  bits of  $S_i$ . Suppose  $S_i(h) = b \in \{0, 1\}$ . It is simple to observe that there is a copy of  $S_i$  starting at  $f_D(\hat{s}_i)$  in  $Y$  if and only if  $S_i(h, L_S) = b, b, \dots, b$  and furthermore, the first undeleted bit after  $S_i$  in  $X$  is also  $b$ . In other words, the  $h$ th bit of  $S_i$  should belong to the final “run” of zeros or ones of  $S_i$  and the first undeleted bit after  $S_i$  should also be of the same value. With probability  $\beta(1 - \beta)^{L_S-1}$ , exactly the  $h$ th bit of  $S_i$  is deleted and with probability  $2^{-(L_S-h+1)}$  the bits after  $h$ th bit in  $S_i$  and the first undeleted bit after  $S_i$  have the same value as the  $h$ th bit of  $S_i$ . The overall probability of this case is  $\beta(1 - \beta)^{L_S-1}2^{-(L_S-h+1)}$ . Similarly, there is a copy of  $S_i$  finishing at  $f_D(\hat{s}_i)$  in  $Y$  if and only if all bits before the  $h$ th bit in  $S_i$  and the first undeleted bit before  $S_i$  are equal to the  $h$ th bit of  $S_i$ . This case happens with probability  $\beta(1 - \beta)^{L_S-1}2^{-h}$ . The intersection of the two events happens when  $S_i$  is all-zeros or all-ones string and the immediate undeleted bits before and after  $S_i$  have the same value as the bits in  $S_i$ . This case happens with probability  $\beta(1 - \beta)^{L_S-1}2^{-(L_S+1)}$ . By using the inclusion exclusion principle and by varying  $h$  from 1 to  $L_S$  we find the total probability of having one deletion within  $S_i$  and a right match for  $S_i$  to be:

$$\begin{aligned} \beta(1 - \beta)^{L_S-1} \sum_{h=1}^{L_S} \left( 2^{-(L_S-h+1)} + 2^{-h} - 2^{-(L_S+1)} \right) &= \\ \beta(1 - \beta)^{L_S-1} (2 - 2^{1-L_S} - L_S 2^{-(L_S+1)}) &= \\ 2\beta + o(\beta), \end{aligned}$$

where in the last step we assumed  $L_S = O(\log \frac{1}{\beta})$ . ■

**Lemma 4.** *With probability  $o(\beta)$ ,  $S_i$  has more than one deletions.*

*Proof:* Since the probability of no deletion within  $S_i$  is  $(1 - \beta)^{L_S}$  and the probability of one deletion within  $S_i$  is  $L_S\beta(1 - \beta)^{L_S-1}$  then the probability of more than one deletion within  $S_i$  is

$$1 - (1 - \beta)^{L_S} - L_S\beta(1 - \beta)^{L_S-1} = o(\beta),$$

where we assumed  $L_S = O(\log \frac{1}{\beta})$  in the final estimate. ■

Let us define  $R := 1 - L_S\beta + 2\beta$ . From the preceding lemmas we conclude that:

**Lemma 5.** *For a random string  $X$  and a random deletion pattern  $D$ , on average the number of pivots with a right match in  $Y$  is  $(R + o(\beta))k$ .*

By applying Theorem 3 we conclude that:

**Lemma 6.** *For a random string  $X$  and a random deletion pattern  $D$ , with probability  $1 - 2^{-\Omega(n)}$ , there are  $(R + o(\beta))k$  pivots with a right match in  $Y$ .*

## B. The Matching Graph

The task of the matching module is to detect right matches of  $S_i$ 's within  $Y$ . For this purpose we use a graph theoretic method. We define a graph  $G(V, E)$  with the vertex set as follows. The graph  $G$  has  $k + 1$  layers of vertices which are denoted by  $\Lambda_0, \Lambda_1, \dots, \Lambda_k$ . Each vertex in layer  $\Lambda_i$ ,  $1 \leq i \leq k - 1$ , represents a match of pivot  $S_i$  in string  $Y$ . We refer to the vertices of  $\Lambda_i$  and matches of  $S_i$  in  $Y$  interchangeably. For vertex  $v \in \Lambda_i$ , let  $\tilde{v}$  and  $\hat{v}$  denote, respectively, the first and the last indices of the match of  $S_i$  corresponding to  $v$  in  $Y$ . We introduce two auxiliary vertices  $s$  and  $t$  where  $\Lambda_0 = \{s\}$  with  $\hat{s} = 0$  and  $\Lambda_k = \{t\}$  with  $\hat{t} = |Y| + 1$ . Vertices  $s$  and  $t$  represent the beginning and the end of string  $Y$ .

We say a vertex in  $\Lambda_i$  is a *good* vertex if it corresponds to a right match of  $S_i$  within  $Y$ . We call a vertex in  $\Lambda_i$  a *bad* vertex if it corresponds to a wrong match of  $S_i$ . By definition of right and wrong matches, in each layer of graph  $G$ , there are possibly zero, one, or two good vertices. In order to detect the right matches of  $S_i$ 's within  $Y$ , we need to find good vertices in graph  $G$ . For that, we define the edge set of  $G$  such that the good vertices are distinguished by their connectivity in the graph.

Let us define the distance between two vertices  $u$  and  $v$  in  $G$  as follows:

$$\text{Dis}(u, v) := \tilde{v} - \hat{u} - 1.$$

Notice that  $\text{Dis}(u, v)$  is nonnegative only when the first bit of  $v$  appears after the last bit of  $u$ . In that case,  $\text{Dis}(u, v)$  is the number of bits between  $u$  and  $v$  in  $Y$ .

For two pivots  $S_i$  and  $S_j$  with  $i < j$ , the number of bits between them in  $X$  is given by

$$(j - i - 1)L_S + (j - i)L_T.$$

If both  $S_i$  and  $S_j$  have right matches in  $Y$ , the number of bits between the right match for  $S_i$  and the right match for  $S_j$  is at most  $(j - i - 1)L_S + (j - i)L_T$ .

Furthermore, in most cases, for  $i < j$ , the first bit of the right match for  $S_j$  appears after the last bit of the right match for  $S_i$ . To see this, notice that, since the first bit of  $S_j$  appears after the last bit of  $S_i$  in  $X$ , if there are no deletions within  $S_i$  and  $S_j$ , their order is preserved in  $Y$ .

However, for instance let  $S_i = 0000$  and  $S_j = 0000$ . Also assume that all bits between  $S_i$  and  $S_j$  are deleted except for a single 0 bit, and that exactly one bit is deleted from  $S_i$  and exactly one bit is deleted from  $S_j$ . In this case, the compound substring of  $Y$  corresponding to  $S_i$  and  $S_j$  and the bits in between them in  $X$  is 0000000, where the first four bits constitute the right match for  $S_i$  and the last four bits constitute the right match for  $S_j$ . As we can observe, the first bit for the right match of  $S_j$  is the last bit for the right match of  $S_i$ . The distance between the right match for  $S_i$  and the right match for  $S_j$  is  $-1$ . It is easy to verify that in general for  $j > i$ , the least value of the distance between the right match of  $S_i$  and the right match of  $S_j$  is  $-1$ .

Based on the two preceding observations, we connect a vertex  $u \in \Lambda_i$  to a vertex  $v \in \Lambda_j$  if and only if

$$-1 \leq \text{Dis}(u, v) \leq (j - i - 1)L_S + (j - i)L_T. \quad (5)$$



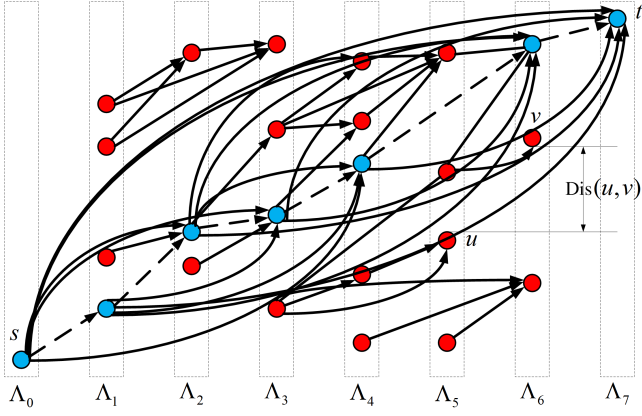


Figure 3. Figure illustrates a graph  $G$  with 8 layers of vertices. The horizontal axis indicate different layers and the vertical axis indicates the position of each vertex in string  $Y$ . The good and bad vertices are distinguished by blue and red colors respectively. The first layer has only one vertex  $s$  and the last layer has only one vertex  $t$ . As it is seen, all good vertices in the graph are connected together and they form an  $s-t$  path which is indicated by the dashed edges in the graph.

Therefore all pairs of good vertices are connected together. By definition,  $s$  and  $t$ , which indicate the beginning and the ending of string  $X$  respectively, are treated as “auxiliary” good vertices. Therefore, good vertices across different layers form an  $s-t$  path in graph  $G$ . However, there are potentially many other pairs of vertices that satisfy the condition of Equation (5) and are connected together. Figure 3 illustrates an instance of graph  $G$  with 8 layers and the connections between vertices.

The following theorem shows that with very high probability bad vertices do not contribute to an  $s-t$  path. That is, any  $s-t$  path of the appropriate length in graph  $G$  is formed mostly of good vertices.

**Theorem 5.** *For a random string  $X$  and a random deletion pattern  $D$ , for  $L_S \geq 11 + 2 \log \frac{1}{\beta}$ , if we pick any path from  $s$  to  $t$  with  $Rk + o(\beta)k$  vertices, then with probability at least  $1 - 2^{-\Omega(n)}$  the path has at least  $Rk - \beta k + o(\beta)k$  good vertices.*

Theorem 5 is not only an existence statement, but also has an algorithmic implication. The implication is that if we pick any path from  $s$  to  $t$  with  $Rk + o(\beta)k$  vertices, the path has many good vertices. Since finding an  $s-t$  path of an appropriate length in  $G$  is a computationally tractable task (we will discuss the computational complexity in the next section), finding a large fraction of good vertices is also a tractable task. Next we prove the theorem.

*Proof:* We begin by finding an upper bound on the probability of the existence of a path  $Q$  from  $s$  to  $t$  with  $Rk + o(\beta)k$  vertices out of which  $\alpha k$  are bad vertices, for some  $\beta \leq \alpha \leq 1$ . There are  $k+1$  layers in graph  $G$  and by Lemma 6 with probability  $1 - 2^{-\Omega(n)}$  there are  $Rk + o(\beta)k$  layers with good vertices in graph  $G$ . Let us fix the realization of the deletion pattern  $D$  and the realization of the pivots  $S_i$  in  $X$  with exactly one deletion and the realization of the immediate undeleted bits before and after pivots  $S_i$  in  $X$  with exactly one deletion. In this way, good vertices of the graph  $G$  are fixed. We consider two cases:

*Case 1:*  $\beta \leq \alpha < \frac{1}{2}$

For  $\beta \leq \alpha < \frac{1}{2}$ , first we fix the layers which have a vertex on the path  $Q$ . Since by assumption, there are  $Rk - \alpha k + o(\beta)k$  good vertices on the path  $Q$ , the selection of good vertices on the path can be done in the following number of ways

$$\binom{Rk + o(\beta)k}{Rk - \alpha k + o(\beta)k} \cdot \binom{(1-R)k + \alpha k + o(\beta)k}{\alpha k} \approx 2^{k((R+o(\beta))H(\frac{\alpha}{R+o(\beta)}) + (1-R+\alpha+o(\beta))H(\frac{\alpha}{1-R+\alpha+o(\beta)}))} = 2^{k(RH(\frac{\alpha}{R}) + (1-R+\alpha)H(\frac{\alpha}{1-R+\alpha}) + o(\beta))} \quad (6)$$

where the first term of the multiplication stands for the number of ways we choose the layers with good vertices on the path  $Q$  and the second term stands for the number of ways we can choose the layers with bad vertices from the remaining available layers.

Suppose that path  $Q$  has vertices from layers  $\Lambda_{i_1}, \Lambda_{i_2}, \dots, \Lambda_{i_{Rk+o(\beta)k}}$ . Let  $\mathcal{I} := \{1, \dots, Rk + o(\beta)k\}$  be the set of indices of the layers with a vertex on the path  $Q$ . Let  $\mathcal{I} = \mathcal{I}_g \cup \mathcal{I}_b$  where  $\mathcal{I}_g$  is the set of indices of layers with good vertices in  $Q$  and  $\mathcal{I}_b$  is the set of indices of layers with bad vertices in  $Q$  (The sets  $\mathcal{I}_g$  and  $\mathcal{I}_b$  are disjoint.). That is, a layer  $\Lambda_{i_j}$  with  $j \in \mathcal{I}_g$  is a layer with a good vertex in  $Q$  and a layer  $\Lambda_{i_j}$  with  $j \in \mathcal{I}_b$  is a layer with a bad vertex in  $Q$ .

Let us express the path  $Q$  as  $s - v_{i_1} - v_{i_2} - \dots - v_{i_{Rk+o(\beta)k}} - t$  where  $v_{i_j} \in \Lambda_{i_j}$ . The path  $Q$  is uniquely identified by the position of the first bit of its vertices,  $(\check{v}_{i_1}, \check{v}_{i_2}, \dots, \check{v}_{i_{Rk+o(\beta)k}})$ . Equivalently, if we know the distance between consecutive vertices  $(\text{Dis}(v_{i_j}, v_{i_{j+1}}) : j \in \mathcal{I})$ , we can uniquely identify the position of each vertex on the path. Therefore, next we count the number of possible values of the distances between consecutive vertices  $(\text{Dis}(v_{i_j}, v_{i_{j+1}}) : j \in \mathcal{I})$ .

Since good vertices are pinned down on the path, the value of  $\text{Dis}(v_{i_j}, v_{i_{j+1}})$  is determined if both  $v_{i_j}$  and  $v_{i_{j+1}}$  are good vertices. Let us define the set  $\mathcal{H} \subset \mathcal{I}$  as follows

$$\mathcal{H} = \{j : j \in \mathcal{I}_b \vee (j+1) \in \mathcal{I}_b\}.$$

Therefore  $(\text{Dis}(v_{i_j}, v_{i_{j+1}}) : j \in \mathcal{H})$  is the set of distances between consecutive vertices of  $Q$  that are undetermined. The number of bad vertices on  $Q$  is  $\alpha k$ . Therefore  $|\mathcal{H}| \leq 2\alpha k$ . Let  $j_1, j_2$  with  $j_1 < j_2$  be two consecutive elements of  $\mathcal{I}_g$ . Then by additivity of distances, bad vertices  $v_{i_{j_1+1}}, \dots, v_{i_{j_2-1}}$  need to satisfy the following constraint:

$$\sum_{t=j_1}^{j_2-1} \text{Dis}(v_{i_t}, v_{i_{t+1}}) = \text{Dis}(v_{i_{j_1}}, v_{i_{j_2}}) - (j_2 - j_1 - 1)L_S, \quad (7)$$

where  $(j_2 - j_1 - 1)L_S$  is the total length of the substrings  $v_{i_{j_1+1}}, \dots, v_{i_{j_2-1}}$  in  $Y$ . Furthermore, bad vertices should be placed on  $Q$  such that they satisfy the constraint given in (5). For every  $j \in \mathcal{H}$ , we need to have

$$-1 \leq \text{Dis}(v_{i_j}, v_{i_{j+1}}) \leq (i_{j+1} - i_j)L_T + (i_{j+1} - i_j - 1)L_S. \quad (8)$$

Next we find an upper bound on the number of integer vectors  $(\text{Dis}(v_{i_j}, v_{i_{j+1}}) : j \in \mathcal{H})$  that satisfy (7) and (8).

For  $j \in \mathcal{I}$  we use the following change of variables

$$\delta_j := (i_{j+1} - i_j)L_T + (i_{j+1} - i_j - 1)L_S - \text{Dis}(v_{i_j}, v_{i_{j+1}}).$$

Equation (7) in terms of the variables  $\delta_j$ 's is written as follows. For  $j_1 < j_2$ , as any two consecutive elements in the ordered version of  $\mathcal{I}_g$ , we have

$$\sum_{j=j_1}^{j_2-1} \delta_j = (i_{j_2} - i_{j_1})L_T + (i_{j_2} - i_{j_1} - 1)L_S - \text{Dis}(v_{i_{j_1}}, v_{i_{j_2}}). \quad (9)$$

Observe that in Equation (9),  $(i_{j_2} - i_{j_1})L_T + (i_{j_2} - i_{j_1} - 1)L_S$  is the number of bits between  $S_{i_{j_1}}$  and  $S_{i_{j_2}}$  in  $X$  and  $\text{Dis}(v_{i_{j_1}}, v_{i_{j_2}})$  is the number of bits between the right match of  $S_{i_{j_1}}$  and the right match of  $S_{i_{j_2}}$  in  $Y$ . Therefore, the right hand side of Equation (9) is the number of deleted bits in the substring between  $S_{i_{j_1}}$  and  $S_{i_{j_2}}$  in  $X$ . To find an upper bound on the number of solutions for  $(\delta_j : j \in \mathcal{H})$ , we relax constraints (9) over all  $j$ 's into a single constraint by adding them together:

$$\sum_{j \in \mathcal{H}} \delta_j = \delta - \sum_{j' \in \mathcal{H}^c} \delta_{j'}. \quad (10)$$

Here  $\delta$  is the total number of deleted bits from  $X$ . The set  $\mathcal{H}^c = \mathcal{I} \setminus \mathcal{H}$  is the set of indices  $j'$  for which  $h_{j'}$  is determined; i.e.,  $v_{i_{j'}}$  and  $v_{i_{j'+1}}$  are both good vertices. Furthermore,  $\delta_{j'}$  is the number of deleted bits from the substring between  $S_{i_{j'}}$  and  $S_{i_{j'+1}}$  in  $X$ .

Next we use the following result on the concentration of  $\sum_{j \in \mathcal{H}} \delta_j$  around its expected value.

**Lemma 7.** *For a random string  $X$  and a random deletion pattern  $D$  the following bound holds:*

$$\Pr \left\{ \left| \sum_{j \in \mathcal{H}} \delta_j - \mathbb{E} \left[ \sum_{j \in \mathcal{H}} \delta_j \right] \right| = o(\beta)k \right\} \geq 1 - 2^{-\Omega(n)}.$$

*Proof:* See Appendix II. ■

To estimate  $\mathbb{E} \left[ \sum_{j \in \mathcal{H}} \delta_j \right]$ , first notice that the average number of deleted bits from  $X$  is  $\mathbb{E}[\delta] = n\beta = (1 + o(\beta))k$ .

Next we find  $\mathbb{E}[\delta_{j'}]$  for  $j' \in \mathcal{H}^c$ . Since  $Q$  has  $Rk + o(\beta)k$  vertices, the average size of the substring between  $S_{i_{j'}}$  and  $S_{i_{j'+1}}$  in  $X$  is  $\frac{n}{Rk + o(\beta)k}$ . Therefore,  $\mathbb{E}[\delta_{j'}]$ , the average number of deleted bits from the substring between  $S_{i_{j'}}$  and  $S_{i_{j'+1}}$  in  $X$  is

$$\mathbb{E}[\delta_{j'}] = \frac{n\beta}{Rk + o(\beta)k} = \frac{1}{R + o(\beta)} = 1 + \beta L_S - 2\beta + o(\beta).$$

Since  $|\mathcal{H}| \leq 2\alpha k$  and  $|\mathcal{I}| = |\mathcal{H}| + |\mathcal{H}^c| = (R + o(\beta))k$ , we find that

$$|\mathcal{H}^c| \geq (R - 2\alpha + o(\beta))k = (1 - \beta L_S + 2\beta - 2\alpha + o(\beta))k.$$

We conclude that

$$\begin{aligned} \mathbb{E} \left[ \sum_{j \in \mathcal{H}} \delta_j \right] &= \mathbb{E}[\delta] - \mathbb{E} \left[ \sum_{j' \in \mathcal{H}^c} \delta_{j'} \right] \\ &= k - |\mathcal{H}^c| \mathbb{E}[\delta_{j'}] + o(\beta)k \\ &\leq k(1 - (1 - \beta L_S + 2\beta - 2\alpha)(1 + \beta L_S - 2\beta) + o(\beta)) \\ &= 2\alpha k(1 + \beta L_S - 2\beta) + o(\beta)k, \end{aligned}$$

and therefore by Lemma 7 with probability at least  $1 - 2^{-\Omega(n)}$

$$\sum_{j \in \mathcal{H}} \delta_j = 2\alpha k(1 + \beta L_S - 2\beta) + o(\beta)k. \quad (11)$$

Therefore, we showed that Equation (7) yields the weaker constraint in (11) on the vector  $(\delta_j : j \in \mathcal{H})$ .

Now consider Inequality (8). We can rewrite it in terms of  $\delta_j$  as follows

$$0 \leq \delta_j \leq (i_{j+1} - i_j)L_T + (i_{j+1} - i_j - 1)L_S + 1.$$

To find an upper bound on the number of solutions for  $(\delta_j : j \in \mathcal{H})$ , we relax the preceding constraint to  $\delta_j \geq 0$ .

Under the constraint that  $\delta_j \geq 0$ , the number of integer solutions for  $(\delta_j : j \in \mathcal{H})$  under Condition (11), is given by

$$\begin{aligned} &\binom{2\alpha k(1 + \beta L_S - 2\beta) + o(\beta)k + |\mathcal{H}| - 1}{|\mathcal{H}| - 1} \leq \\ &\binom{2\alpha k(2 + \beta L_S - 2\beta + \frac{o(\beta)}{\alpha})}{2\alpha k} \leq \\ &2^{2\alpha k(2 + \beta L_S - 2\beta + \frac{o(\beta)}{\alpha})} \leq 2^{5\alpha k}, \quad (12) \end{aligned}$$

where the last estimate holds for sufficiently small  $\beta$ .

Given the number of possibilities for path  $Q$ , we next compute the probability of occurrence of each realization of path  $Q$ . Since  $X$  is generated by an i.i.d Bernoulli source of parameter  $\frac{1}{2}$ , different substrings of  $X$  are independent and the probability of any given realization of bad vertices as specified by the choice of  $\delta_j$ 's is  $2^{-L_S \alpha k}$ . By applying the union bound on the probability of existence of individual paths, using Inequality (6) and Inequality (12), we conclude that the probability of the existence of a path  $Q$  with  $Rk + o(\beta)k$  total vertices and  $\alpha k$  bad vertices is upperbounded by  $2^{\Delta_\alpha k}$  where

$$\begin{aligned} \Delta_\alpha &= RH \left( \frac{\alpha}{R} \right) + (1 - R + \alpha)H \left( \frac{\alpha}{1 - R + \alpha} \right) \\ &\quad + 5\alpha - \alpha L_S + o(\beta) \\ &= -\alpha \log \alpha + \alpha \log R - R \log(1 - \frac{\alpha}{R}) \\ &\quad + \alpha \log(1 - \frac{\alpha}{R}) - \alpha \log \alpha + \alpha \log(1 - R + \alpha) \\ &\quad - (1 - R) \log(1 - \frac{\alpha}{1 - R + \alpha}) + 5\alpha - \alpha L_S + o(\beta). \end{aligned}$$

Next we find an upper bound for  $\Delta_\alpha$ . Since  $R < 1$ , then  $\alpha \log R < 0$ . Since  $\alpha < \frac{1}{2}$ , for small enough  $\beta$ ,  $R > \alpha$ . Therefore  $\alpha \log(1 - \frac{\alpha}{R}) < 0$  and  $\alpha \log(1 - R + \alpha) < 0$ . Using the inequality  $\log(1 + x) \leq \frac{x}{\ln 2}$  for  $|x| < 1$  we find that

$$-R \log(1 - \frac{\alpha}{R}) = R \log(1 + \frac{\alpha}{R - \alpha}) \leq \frac{R\alpha}{(R - \alpha) \ln 2} \leq \frac{2\alpha}{\ln 2}$$

where we used the fact that for small values of  $\beta$ ,  $R$  is close to 1 and  $R - \alpha > \frac{1}{2}$ . Also we have

$$\begin{aligned} -(1 - R) \log \left( 1 - \frac{\alpha}{1 - R + \alpha} \right) &= (1 - R) \log \left( 1 + \frac{\alpha}{1 - R} \right) \\ &\leq \frac{(1 - R)\alpha}{(1 - R) \ln 2} = \frac{\alpha}{\ln 2}. \end{aligned}$$



Therefore

$$\begin{aligned}\Delta_\alpha &\leq o(\beta) - 2\alpha \log \alpha + \frac{2\alpha}{\ln 2} + \frac{\alpha}{\ln 2} + 5\alpha - \alpha L_S \\ &= \alpha \left( \frac{o(\beta)}{\alpha} - 2\log \alpha + \frac{3}{\ln 2} + 5 - L_S \right) \\ &< \alpha(-2\log \alpha + 10 - L_S),\end{aligned}$$

where we used  $\frac{o(\beta)}{\alpha} \leq \frac{o(\beta)}{\beta} \rightarrow 0$  as  $\beta \rightarrow 0$ .

Case 2:  $\frac{1}{2} \leq \alpha \leq R + o(\beta)$

We again seek to bound the probability of the existence of a path  $Q$  from  $s$  to  $t$  with  $Rk + o(\beta)k$  total vertices and  $\alpha k$  bad vertices. Let the path  $Q$  be denoted by  $s - v_{i_1} - v_{i_2} - \dots - v_{i_{Rk+o(\beta)k}} - t$  and let  $\delta_j$  denote the number of deleted bits between vertices  $v_{i_j}$  and  $v_{i_{j+1}}$ . Clearly, the sum of  $\delta_j$  is the total number of deletions in string  $Y$ . By Theorem 3, with probability at least  $1 - 2^{-\Omega(n)}$  we have  $\sum_{j=0}^{Rk+o(\beta)k} \delta_j = n\beta + n\beta o(\beta) = k(1 + o(\beta))$ . The number of integer solutions for  $\delta_j \geq 0$  under this constraint is

$$\binom{k + Rk + o(\beta)k - 1}{Rk + o(\beta)k - 1} \leq \binom{2k}{k} \leq 2^{2k}.$$

The probability for each solution of  $\delta_j$ 's to represent a valid  $s-t$  path is at most  $2^{-L_S \alpha k}$ . Therefore, an upper bound on the probability of existence of a path  $Q$  in this case is  $2^{(2-L_S \alpha)k}$ .

Finally, putting both cases for the range of  $\alpha$  together, the probability of the existence of a path  $Q$  with  $Rk + o(\beta)k$  vertices between  $s$  and  $t$  with at least  $\beta k$  bad vertices can be upper bounded by the sum of two integrals:

$$\begin{aligned}&\int_{\alpha=\beta}^{\frac{1}{2}} 2^{\Delta_\alpha k} d\alpha k + \int_{\alpha=\frac{1}{2}}^{R+o(\beta)} 2^{(2-L_S \alpha)k} d\alpha k \leq \\ &\int_{\alpha=\beta}^{\frac{1}{2}} 2^{(-2\log \alpha + 10 - L_S)\alpha k} d\alpha k + \int_{\alpha=\frac{1}{2}}^{R+o(\beta)} 2^{(2-L_S \alpha)k} d\alpha k.\end{aligned}$$

If we pick  $L_S \geq 11 + 2\log \frac{1}{\beta}$  then we find

$$(-2\log \alpha + 10 - L_S)\alpha k \leq -\alpha k \leq -\beta k$$

for  $\beta \leq \alpha \leq \frac{1}{2}$ . Also,

$$2 - L_S \alpha \leq 2 - \frac{1}{2} \cdot 11 = -3.5.$$

Therefore, we can upper bound the sum of the two integrals by

$$\begin{aligned}\int_{\beta}^{\frac{1}{2}} k 2^{-\beta k} d\alpha + \int_{\frac{1}{2}}^{R+o(\beta)} k 2^{-3.5k} d\alpha &\leq \frac{k}{2} (2^{-\beta k} + 2^{-3.5k}) \\ &= 2^{-\Omega(n)}.\end{aligned}$$

This yields the result.  $\blacksquare$

In order to verify the result of Theorem 5 in a practical setting, we have plotted graph  $G$  for randomly generated string  $X$  and randomly generated deletion pattern  $D$  with parameter  $\beta = 0.01$ , for three values of  $L_S$  in Figure 4. To avoid visual complications, we have only plotted edges that connect vertices on two consecutive layers. As it is clear from the figure, for small values of  $L_S$ , there are many edges in the graph and there are potentially many paths that connect

$s$  to  $t$  which do not share many vertices with the correct path. However, for larger values of  $L_S$ , the irrelevant edges disappear from the graph and the only path that remains is the one formed by good nodes of the graph. For  $\beta = 0.01$ , Theorem 5 states that  $L_S \geq 11 + 2\log \frac{1}{\beta} \approx 17$  is sufficient for our purpose. In practice, we observe values of  $L_S$  around 8 are sufficient for distinguishing good vertices on graph  $G$ .

#### IV. PRACTICAL IMPLEMENTATION

In this section we discuss practical implementation of our synchronization protocol, consisting of a matching module, a deletion recovery module, and an LDPC decoder module (see Figure 1).

For the deletion recovery module, we can implement the synchronization protocol of Venkataramanan *et al.* [8] which runs in linear time in  $|P_j|$  for deletion recovery of each substring  $P_j, 1 \leq j \leq k'$ . Therefore the overall complexity of the deletion recovery module is linear in  $n$ . For the LDPC decoder module there are many sophisticated encoding and decoding schemes (see [16], [15]) that need running time linear in  $n$ .

In this section we therefore focus on the implementation of the graph-based algorithm for the matching module explained in the previous section. The result of Theorem 5 indicates that to find a large number of right matches for pivots in the received string  $Y$ , it suffices to find an  $s-t$  path with  $Rk + o(\beta)k$  vertices in the matching graph  $G$ . We now argue that this problem can be cast as the well known "shortest path problem" in a directed graph, so it can be efficiently solved in polynomial time.

As the first step, we only keep the vertices in graph  $G$  which have an edge to vertex  $t$  and remove all other vertices. Since all good vertices are connected to vertex  $t$ , this step does not eliminate any good vertex from graph  $G$ . Let  $\tilde{G}$  denote the resulting graph. As the second step, we find the longest  $s-t$  path in  $\tilde{G}$ . Since all good vertices are connected together and form an  $s-t$  path of length  $Rk + o(\beta)k$ , the longest path in  $\tilde{G}$  has at least  $Rk + o(\beta)k$  vertices. Finally, we modify the discovered path into a path with only  $Rk + o(\beta)k$  vertices by keeping only the first  $Rk + o(\beta)k$  vertices on the path. Since each vertex in the graph  $\tilde{G}$  has an edge to vertex  $t$ , the resulting vertices from this step form a path with  $Rk + o(\beta)k$  vertices from  $s$  to  $t$ .

The only step of the above procedure which is computationally demanding is the second step for finding the longest  $s-t$  path in  $\tilde{G}$ . Notice that since  $G$  and hence  $\tilde{G}$  are acyclic graphs, the longest  $s-t$  path problem in  $\tilde{G}$  can be reduced to the shortest  $s-t$  path problem in  $\tilde{G}$  by assigning weight  $-1$  to each edge. The latter problem is solvable in time  $O(|\tilde{G}|^2)$ , for instance by Dijkstra's algorithm [18], where  $|\tilde{G}|$  is the number of vertices in  $\tilde{G}$ . We upper bound  $|\tilde{G}|$  by  $|G|$ . To approximate  $|G|$ , we notice that there are approximately  $n\beta$  layers in graph  $G$  and the number of vertices in layer  $\Lambda_i$  is the number of copies of pivot  $S_i$  in  $Y$ , which is approximately  $2^{-L_S}|Y| = O(\beta^2 n)$ . Therefore,  $|G| \approx O((\beta^2 n) \cdot (n\beta)) = O(n^2 \beta^3)$ . We conclude that the complexity of matching pivots in graph  $G$  is upper bounded by  $O(|G|^2) = O(n^4 \beta^6)$ .

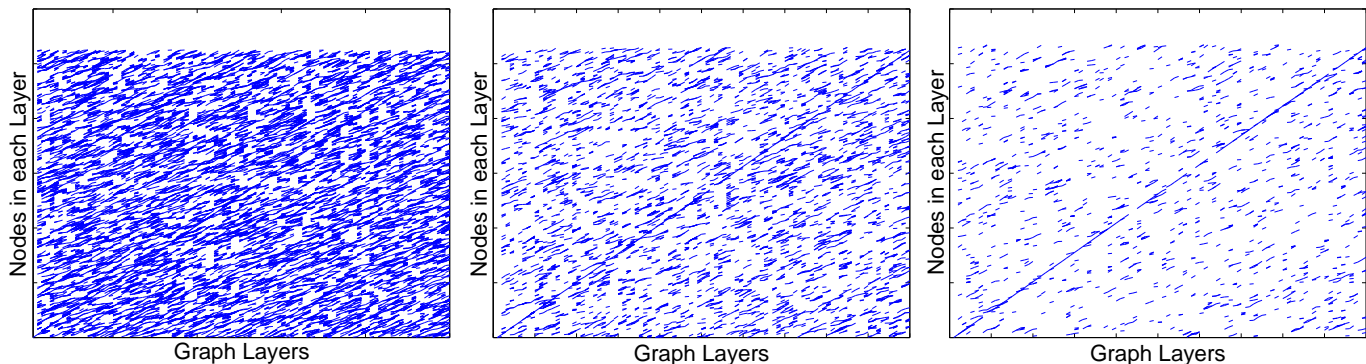


Figure 4. Graph  $G$  for  $k = 100$ ,  $\beta = 0.01$ ,  $L_T = 100$ , and  $L_S = 6, 7$ , and  $8$ , where only the edges between consecutive layers are depicted.

## V. CONCLUSIONS

In this paper we offered the first synchronization protocol for recovering from a small rate of deletions with an optimal order of transmitted bits. The main idea was to divide the synchronization problem into synchronization between shorter substrings of the source file and destination file. For that, our protocol sends equally spaced small substrings of the source file to the destination, and destination then uses a graph theoretic algorithm to locate the short substrings within its file with high accuracy. For synchronization between the shorter substrings we used existing protocols that recover from a small number of edits. We observed that the compound output of the first two steps can be modeled as an output of a BSC with a small error probability. This error can be recovered with a low bit error rate by using an LDPC coding scheme.

While in this work we only considered recovering from i.i.d patterns of deleted bits, there are many other interesting edit models that the ideas of this paper can be applied to. An immediate extension of our work is to the synchronization from i.i.d insertions. To explain an i.i.d insertion process, let us consider an equivalent description of the i.i.d deletion process considered in this paper. In the new description, the deletion pattern  $D$  is described as an independent sequence of positive integers, where the integers alternatively represent the length of zero and one runs in the deletion pattern  $D$ . It is easy to verify that if the integers are generated independently according to an appropriate geometric distribution, the result is an i.i.d 0-1 deletion pattern. We can describe the insertion pattern in the same way by generating the run length sequence of the pattern. For the insertion pattern, each run of ones corresponds to an inserted substring of equal length generated by an i.i.d Bernoulli process. Also, each run of zeros correspond to a substring of the input string of equal length in the output. It is not hard to see that the solution of this paper for synchronization from deletions is directly applicable to solving the synchronization problem from random insertions.

One can also consider more general patterns of deletions or insertions, e.g., the 0-1 deletion (insertion) patterns that follow a markov chain random process (see [12]).

Another interesting direction for the extension of this work is the design of synchronization protocols that are capable of recovering from a small rate of both deletions and insertions.

While the deletion recovery module in our work, based on the algorithm by Venkataramanan *et al.* [8], is directly applicable to recovery from deletions and insertions, the main challenge is to extend the graph theoretic algorithm for matching the pivot substrings in the received string  $Y$  when there are both deletions and insertions. Again, many parts of our argument still hold for the new setting as long as the edits happen with small rates while some technical parts may need to be modified. This extension is the focus of our current research.

There are some other aspects of our current research that can be modified into more efficient synchronization protocols. For example, our algorithm needs a small backward bandwidth from node  $B$  to node  $A$  in the deletion recovery module. This bandwidth is an inherent component of the synchronization protocol of Venkataramanan *et al.*, [8]. It is of great interest to design protocols that can operate on forward links only. As proved by Orlitsky [3], design of optimal protocols for recovery from deletions on forward links implies optimal protocols for recovery from deletions and insertions. Furthermore, such protocols can be implemented as efficient channel codes for communicating over edit channels [12], [19], [20], [21].

Finally, from a practical perspective, it is interesting to design a more efficient implementation of the graph theoretic matching algorithm which is at the heart of our matching module. While our algorithm runs in  $O(n^4\beta^6)$  time, we believe that by exploiting the specific structure of the matching graph, and applying additional restrictions on the connectivity of the vertices of the graph together, it is possible to considerably reduce the running time of the matching module and hence reduce the overall complexity of the synchronization protocol.

## ACKNOWLEDGEMENT

The work is supported in part by NSF-CCF grant no. CAREER-1150212.

## REFERENCES

- [1] R. R. Varshamov and G. M. Tenegolts, "Codes which correct single asymmetric errors (in Russian)," *Avtomatika i Telemekhanika*, vol. 26, pp. 286–290, 1965.
- [2] V. L. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals (in Russian)," *Soviet Physics Dokl.*, vol. 10, pp. 707–710, 1966.

- [3] A. Orlitsky, "Interactive communication of balanced distributions and of correlated files," *SIAM J. Discret. Math.*, vol. 6, no. 4, pp. 548–564, 1993.
- [4] T. Schwarz, R. W. Bowdidge, and W. A. Burkhard, "Low cost comparison of file copies," in *Proc. of the 10th Int. Conf. on Distributed Computing Systems*, 1990, pp. 196–202.
- [5] G. Cormode, M. Paterson, S. C. Sahinalp, and U. Vishkin, "Communication complexity of document exchange," in *Proc. of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Jan. 2000, pp. 197–206.
- [6] A. V. Evfimievski, "A probabilistic algorithm for updating files over a communication link," in *Proc. of the 9th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Jan. 1998, pp. 300–305.
- [7] A. Orlitsky and K. Viswanathan, "Practical protocols for interactive communication," in *Proc. of IEEE International Symposium on Information Theory (ISIT)*, Jun. 2001, p. 115.
- [8] R. Venkataramanan, H. Zhang, and K. Ramchandran, "Interactive low-complexity codes for synchronization from deletions and insertions," in *Proc. of the 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sep.-Oct. 2010, pp. 1412–1419.
- [9] A. Tridgell, "Efficient algorithms for sorting and synchronization," Ph.D. dissertation, Australian National University, 2000.
- [10] T. Suel, P. Noel, and D. Trendafilov, "Improved file synchronization techniques for maintaining large replicated collections over slow networks," in *Proc. of the 20th International Conference on Data Engineering (ICDE)*, Mar.-Apr. 2004, pp. 153–164.
- [11] H. Zhang, C. Yeo, and K. Ramchandran, "VSYNC: a novel video file synchronization protocol," in *Proc. of the 16th ACM International Conference on Multimedia*, Jun. 2008, pp. 757–760.
- [12] N. Ma, K. Ramchandran, and D. Tse, "Efficient file synchronization: a distributed source coding approach," in *Proc. of the IEEE International Symposium on Information Theory (ISIT)*, Jul.-Aug. 2011, pp. 583–587.
- [13] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, vol. 19, no. 4, pp. 471–480, Jul. 1973.
- [14] A. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Transactions on Information Theory*, vol. 22, no. 1, pp. 1–10, Jan. 1976.
- [15] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.
- [16] —, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.
- [17] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, Mar. 1963.
- [18] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, Jan. 1959.
- [19] Y. Kanoria and A. Montanari, "On the deletion channel with small deletion probability," in *Proc. of IEEE International Symposium on Information Theory (ISIT)*, Jun. 2010, pp. 1002–1006.
- [20] A. Kalai, M. Mitzenmacher, and M. Sudan, "Tight asymptotic bounds for the deletion channel with small deletion probabilities," in *Proc. of IEEE International Symposium on Information Theory (ISIT)*, Jun. 2010, pp. 997–1001.
- [21] M. Mitzenmacher, "A survey of results for deletion channels and related synchronization channels," *Probability Surveys*, vol. 6, pp. 1–33, 2009.

## APPENDIX I

Here we evaluate  $\mathbb{E}[\delta_j \log |P_j|]$ .

$$\begin{aligned}
 \mathbb{E}[\delta_j \log |P_j|] &= \sum_l \Pr\{|P_j| = l\} \mathbb{E}[\delta_j \log |P_j| | |P_j| = l] \\
 &= \sum_l \beta l \log l \Pr\{|P_j| = l\} \\
 &= \mathbb{E}[\beta |P_j| \log |P_j|].
 \end{aligned}$$

Next we estimate  $\mathbb{E}[\beta |P_j| \log |P_j|]$ . Notice that  $P_j$  is the substring of  $X$  between  $S_{i_{j-1}}$  and  $S_{i_j}$ . There are  $(i_j - i_{j-1})$  data strings and  $(i_j - i_{j-1} - 1)$  pivot strings between  $S_{i_{j-1}}$

and  $S_{i_j}$ . Therefore

$$|P_j| = (i_j - i_{j-1})L_T + (i_j - i_{j-1} - 1)L_S.$$

There is a total of  $k$  pivots, and  $k'$  of them are matched by the matching module. Therefore, with probability  $p := \frac{k'}{k} \approx (1 - L_S\beta + 2\beta)$  a pivot  $S_i$  is matched. Furthermore, the probability that a pivot is matched is independent of other pivots. Thus,  $(i_j - i_{j-1})$  has the following geometric distribution

$$\Pr\{i_j - i_{j-1} = r\} = p(1 - p)^{r-1}.$$

Suppose  $r \in \{1, 2, \dots\}$  is a random variable distributed as above. If we upper bound  $|P_j| \leq (i_j - i_{j-1})(L_T + L_S)$ , then

$$\begin{aligned}
 \mathbb{E}[\beta |P_j| \log |P_j|] &\leq \mathbb{E}[\beta r(L_T + L_S) \log r(L_T + L_S)] \\
 &= \beta(L_T + L_S) \mathbb{E}[r \log r + r \log(L_T + L_S)] \\
 &\leq 2\mathbb{E}[r^2] + 2\log(L_T + L_S) \mathbb{E}[r],
 \end{aligned}$$

where we used the fact that  $\beta(L_T + L_S) \leq 2$  and  $r \log r \leq r^2$ . We can write

$$\mathbb{E}[r] = \frac{1}{p}, \mathbb{E}[r^2] = \text{Var}(r) + \mathbb{E}[r]^2 = \frac{2-p}{p^2}.$$

Also, we use  $\log(L_T + L_S) \leq \log 2L_T \leq 2\log \frac{1}{\beta}$  and find that

$$\mathbb{E}[\beta |P_j| \log |P_j|] \leq \frac{4-2p}{p^2} + \frac{4}{p} \log \frac{1}{\beta} \leq 16 + 8 \log \frac{1}{\beta},$$

where we used the fact that  $\frac{4-2p}{p^2} \leq 16$  for  $p \geq \frac{1}{2}$  (Notice that  $p \rightarrow 1$ , as  $\beta \rightarrow 0$ ).

## APPENDIX II

Recall that  $\delta_j$  is the number of deleted bits from the substring of  $X$  between pivots  $S_{i_j}$  and  $S_{i_{j+1}}$ . Let us denote by  $\mathcal{L}_S$  the set of indices  $l$  for which  $S_l$  appears between  $S_{i_j}$  and  $S_{i_{j+1}}$  for some  $j \in \mathcal{H}$ . Similarly, let  $\mathcal{L}_T$  denote the set of indices  $l$  for which  $T_l$  appears between  $S_{i_j}$  and  $S_{i_{j+1}}$  for some  $j \in \mathcal{H}$ . Let  $\delta_{S_l}$  denote the number of deleted bits from  $S_l$  and  $\delta_{T_l}$  denote the number of deleted bits from  $T_l$ . We can write

$$\sum_{j \in \mathcal{H}} \delta_j = \sum_{l \in \mathcal{L}_S} \delta_{S_l} + \sum_{l \in \mathcal{L}_T} \delta_{T_l}.$$

Notice that the length of the interval that  $\delta_{S_l}$  takes values from is  $L_S = O(\log \frac{1}{\beta})$  and the length of the interval that  $\delta_{T_l}$  takes values from is  $L_T = \frac{1}{\beta}$ . Next, by application of Theorem 4

we can write

where in our derivation we used the fact that  $|\mathcal{L}_S| \leq k$  and  $|\mathcal{L}_T| \leq k$ , since  $\mathcal{L}_S$  and  $\mathcal{L}_T$  are subsets of  $\{1, \dots, k-1\}$ .

$$\begin{aligned}
& \Pr \left\{ \left| \sum_{j \in \mathcal{H}} \delta_j - \mathbb{E} \left[ \sum_{j \in \mathcal{H}} \delta_j \right] \right| = o(\beta)k \right\} \geq \\
& \Pr \left\{ \left| \sum_{l \in \mathcal{L}_S} \delta_{S_l} - \mathbb{E} \left[ \sum_{l \in \mathcal{L}_S} \delta_{S_l} \right] \right| + \left| \sum_{l \in \mathcal{L}_T} \delta_{T_l} - \mathbb{E} \left[ \sum_{l \in \mathcal{L}_T} \delta_{T_l} \right] \right| = o(\beta)k \right\} = \\
& \Pr \left\{ \left| \sum_{l \in \mathcal{L}_S} \delta_{S_l} - \mathbb{E} \left[ \sum_{l \in \mathcal{L}_S} \delta_{S_l} \right] \right| = o(\beta)k \right\} \cdot \\
& \quad \Pr \left\{ \left| \sum_{l \in \mathcal{L}_T} \delta_{T_l} - \mathbb{E} \left[ \sum_{l \in \mathcal{L}_T} \delta_{T_l} \right] \right| = o(\beta)k \right\} = \\
& \Pr \left\{ \left| \sum_{l \in \mathcal{L}_S} \delta_{S_l} - \mathbb{E} \left[ \sum_{l \in \mathcal{L}_S} \delta_{S_l} \right] \right| = \frac{o(\beta)k}{|\mathcal{L}_S|} |\mathcal{L}_S| \right\} \cdot \\
& \quad \Pr \left\{ \left| \sum_{l \in \mathcal{L}_T} \delta_{T_l} - \mathbb{E} \left[ \sum_{l \in \mathcal{L}_T} \delta_{T_l} \right] \right| = \frac{o(\beta)k}{|\mathcal{L}_T|} |\mathcal{L}_T| \right\} \geq \\
& \left( 1 - 2 \exp\left(-\frac{2o(\beta^2)k^2|\mathcal{L}_S|}{|\mathcal{L}_S|^2 L_S^2}\right) \right) \cdot \left( 1 - 2 \exp\left(-\frac{2o(\beta^2)k^2|\mathcal{L}_T|}{|\mathcal{L}_T|^2 L_T^2}\right) \right) = \\
& \left( 1 - 2 \exp\left(-\frac{2o(\beta^2)k^2}{|\mathcal{L}_S| L_S^2}\right) \right) \cdot \left( 1 - 2 \exp\left(-\frac{2o(\beta^2)k^2}{|\mathcal{L}_T| L_T^2}\right) \right) \geq \\
& \left( 1 - 2 \exp\left(-\frac{2o(\beta^2)k}{L_S^2}\right) \right) \cdot \left( 1 - 2 \exp\left(-\frac{2o(\beta^2)k}{L_T^2}\right) \right) \geq \\
& \left( 1 - 2 \exp\left(-\frac{2o(\beta^2)\beta}{O(\log^2 \frac{1}{\beta})} n\right) \right) \cdot \left( 1 - 2 \exp(-2\beta^3 o(\beta^2)n) \right) = \\
& (1 - 2^{-\Omega(n)}) \cdot (1 - 2^{-\Omega(n)}) = 1 - 2^{-\Omega(n)},
\end{aligned}$$